INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

Automated Time Domain Modeling of Linear and Nonlinear Microwave Circuits Using Recurrent Neural Networks

by

Hitaish Sharma, B.A.Sc.,

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirement for the
degree of Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Electronics
Carleton University
Ottawa, Ontario K1S 5B6
Canada

©Copyright September 2005, Hitaish Sharma

Library and
Archives Canada

Bibliothèque et
Archives Canada

0-494-08385-9

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file *Votre référence*

ISBN:

Our file *Notre référence*

ISBN:

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Q. J. Zhang, for his professional and research advice, financial support, and guidance during the research and the preparation of this thesis. I am fortunate to have the opportunity to learn from and work with such a leading figure in the scientific community, and know that it will benefit me for the rest of my life.

I also want to thank all my fellow students and researchers within the research group for making the last two years an enlightening experience. Special thanks to Dr. Jianjun Xu, Lei Zhang, Yi Cao, Larry Ton, Nabil Yazdani, and Humayun Kabir for making the lab a fun place.

Finally, I would like to thank my parents. Without them I would not be here to write this thesis.

Contents

Chapter 1 Introduction	1
1.1 Thesis Motivation	1
1.2 Thesis Objective	3
1.3 Thesis Organization	4
 Chapter 2 Literature Review	 6
2.1 Artificial Neural Networks for RF/Microwave Design.....	6
2.2 Neural Network Structures.....	9
2.2.1 Multilayer Perceptron.....	11
2.2.2 Neural Networks with Feedback.....	18
2.2.2.1 Dynamic Neural Network.....	20
2.2.2.2 Recurrent Neural Network.....	24
2.3 Automatic Neural Model Generation.....	28
2.4 Conclusions.....	33
 Chapter 3 Automatic RNN Modeling.....	 34
3.1 Introduction.....	34
3.2 RNN Macromodel.....	35
3.3 AMG for RNN.....	37
3.4 Summary.....	44

Chapter 4 Transient EM Modeling Using RNN.....	48
4.1 Introduction.....	48
4.2 RNN Training with Transient EM Data.....	49
4.3 Circuit Simulator Implementation of RNN Macromodel.....	52
4.4 WR-28 Waveguide Example.....	54
4.5 Microstrip Filter Example.....	56
 Chapter 5 RNN Behavioral Modeling of Power Amplifiers	63
5.1 Introduction.....	63
5.2 Power Amplifier Envelope Model.....	64
5.3 RFIC Power Amplifier Example.....	66
 Chapter 6 Conclusions and Future Research	76
6.1 Conclusions	76
6.2 Suggestions for Future Research	77
 Bibliography	79

List of Figures

Figure 2.1: Three-layer feedforward neural network (FFNN) structure with n inputs, h hidden layer neurons, and m outputs	10
Figure 2.2: Information processing within the j^{th} hidden layer neuron of the 3 layer MLP (MLP3). The MLP3 has n inputs and m outputs.	12
Figure 2.3: Neuron activation functions for hidden layer neurons of the MLP. All the functions are bounded, continuous, monotonic, and continuously differentiable.	14
Figure 2.4: Summary of steps to train the MLP. In step 5, many gradient-based optimization algorithms such as back-propagation, steepest descent, conjugate gradient, and quasi-Newton can be used to determine the weight update.	17
Figure 2.5: a) 1-input and 1-output static MLP3 to be trained with discrete samples from an input-output TD sequence. b) Training data distribution of the samples of the input-output waveform. A static model will not be able to handle multiple outputs for the same input.	19
Figure 2.6: DNN model based on the MLP for a SISO system (single-input, single-output). The derivative information in the input allows for TD modeling.....	22
Figure 2.7: Circuit implementation of the DNN model. The state variables (v) are voltages across unit capacitances ($C=1F$) while the input (u) is the current through unit inductances ($L=1H$).....	23

Figure 2.8: Training of the DNN using the input and output spectrum of a nonlinear microwave circuit. Once successfully trained, the DNN can be used in circuit simulators as a fast and accurate model of the entire nonlinear circuit.	25
Figure 2.9: Training of the RNN using TD input-output sequences from a nonlinear microwave circuit. Note that the previous outputs (feedback) plus input history is used to determine the next RNN value..	26
Figure 2.10: a) Training ($\bullet$) and validation ($\times$) data in a subregion of two-dimensional input space. b) After training, if the validation sample ($\times$) has the highest error among the entire validation data, the sub-region is considered the worst and is then subdivided (star-distribution) to generate new training (P) and validation (Q) samples. Note that the original validation sample is now a training sample in the next stage ($\otimes$).	31
Figure 2.11: Flowchart of the AMG algorithm to train a NN structure (S) over k stages. Both automatic data generation and NN training are combined so that a good neural model can be achieved. If some of the training data contains large errors (measurement or accidental errors), Huber quasi-Newton training is performed to ignore the errors..	32
Figure 3.1: RNN structure with output feedback (M_y). The RNN is a discrete-time structure trained with sampled input-output data..	36
Figure 3.2: Relationship between RNN and RNN-trainer. The RNN-trainer structure uses only the parameters (p) to generate the entire output training waveform by sweeping the index (k) from 0 to N_T-1 (# samples = N_T).	39

Figure 3.3: Flowchart showing the process to achieve good training of a RNN model.	
AMG automates the process by using the RNN-trainer structure.	45
Figure 3.4: Flowchart showing how AMG attempts to reduce the final RNN-trainer structure to a more compact model.	46
Figure 4.1: EM simulation setup for EM data generation.....	50
Figure 4.2: Top view of WR-28 waveguide with dimensions d between conducting posts.	55
Figure 4.3: Comparison between waveguide RNN responses (-) and TLM responses (■) for various d . In the f_{21} response, an initial output delay has been removed before training.....	57
Figure 4.4: 2-port frequency responses of RNN sub-circuit for various d of the WR-28 waveguide example.	58
Figure 4.5: Microstrip filter with dimension L	59
Figure 4.6: Comparison between microstrip RNN responses (-) and TLM responses (■) for different L	61
Figure 4.7: Frequency responses of 2-port RNN sub-circuit for $L=12\text{mm}$, $L=14\text{mm}$, $L=16\text{mm}$ of the microstrip example.	62
Figure 5.1: PA envelope behavioral model for input ($x(t)$) and transmitted signal ($y(t)$). .	65
Figure 5.2: PA envelope behavioral model using RNN. Each RNN learns the nonlinear functions, K_1 and K_2 from (5.1).	67

Figure 5.3: RNN training results. (a) $I_{out}(t)$ comparison between <i>Agilent-ADS</i> and K_1 RNN. (b) $Q_{out}(t)$ comparison between <i>Agilent-ADS</i> and K_2 RNN.	69
Figure 5.4: RNN validation. (a) $I_{out}(t)$ comparison between <i>Agilent-ADS</i> and K_1 RNN for NADC signal. (b) $Q_{out}(t)$ comparison between <i>Agilent-ADS</i> and K_2 RNN for CDMA-2000 signal.....	70
Figure 5.5: AM/AM distortion between simulation and RNN PA behavioral model for 3G WCDMA training sequence. Note the gain variation due to the PA memory effects. (The low Pin point can be better matched with additional training at low power).	72
Figure 5.6: AM/PM distortion between simulation and RNN PA behavioral model for 3G WCDMA training sequence. This nonlinear distortion is important to model because of the impact on phase-shift type modulation schemes.	73
Figure 5.7: Spectral re-growth of RFIC PA for the 3G WCDMA training sequence (chip rate = 3.84 MHz). The RNN PA model accurately matches the circuit simulation results.	74

List of Tables

Table 3-I: AMG Detection of RNN-trainer Underlearning and Overlearning after n th training stage	42
Table 3-II: Comparison between Automatic RNN Modeling and AMG	47
Table 4-I: Transient Excitation Waveforms for Generating RNN Training Data	51
Table 4-II: RNN Training Results for Microstrip Filter Example	60
Table 5-I: RNN Training Results for RFIC PA Example	68
Table 5-II: RNN Validation.....	68
Table 5-III: RFIC PA Model Comparison for In-phase (K_1) Relationship.....	71

Abstract

In this thesis, the recurrent neural network (RNN) is employed as a dynamic time-domain (TD) model for both linear and nonlinear microwave circuits. An automated RNN modeling technique is proposed to efficiently determine the training waveform distribution and internal RNN structure during the offline training process. This technique is an expansion of the existing automatic model generation (AMG) algorithm to support dynamic TD modeling. The automated process is used to train RNN with transient electromagnetic (EM) behavior of microwave structures for varying material and geometrical parameters. TD EM simulators are automatically driven by AMG in the appropriate manner to generate the necessary RNN training waveforms. AMG then varies the RNN structural parameters during training to learn the transient behavior with minimum RNN order while satisfying accuracy requirements. Once trained, the RNN macromodel is inserted into circuit simulators for use in circuit analysis. Automatic RNN modeling is also applied to model nonlinear power amplifier (PA) behavior. An envelope formulation is used to specifically learn the AM/AM and AM/PM distortions due to third-generation (3G) digital modulation input. The RNN PA model is able to model these TD distortions after training and can accurately model the amplifier behavior in both time (AM/AM, AM/PM) and frequency (spectral re-growth).

Chapter 1

Introduction

1.1 Thesis Motivation

As the frequencies and level of integration increase in radio frequency (RF) and microwave circuit design, the need for fast, accurate, and compact models grow. Such models are critical in computer-aided design (CAD) or electronic design automation (EDA) software tools so that the designer can accurately simulate the behavior in a short time. If the models used within the design process are good, the resulting hardware should behave as expected and the overall design cycle can be reduced. The financial benefits of “faster time to market” and shorter design cycle continue to fuel the search for better models as the electronics industry and technology evolves.

Recently, artificial neural networks (NN) have been introduced as potential model candidates in RF/microwave design [1], [2]. These biologically inspired information processing systems are capable of “learning” any multi-dimensional nonlinear input-output relationship to any desired accuracy. When trained with appropriate measurement and/or simulation data, the NN are able to generalize the correct behavior making them useful as models in EDA tools. Another important feature is that NN rely on simple and fast mathematical computations that are much more CPU efficient than physics-based or

electromagnetic (EM)-based models. Since NN models are both accurate and fast, they are ideal for highly repetitive tasks such as statistical analysis and optimization [3]. As well, NN are a very compact solution for multi-dimensional problems when compared with other modeling techniques such as the table look-up method [4].

A major issue in NN model development is the training process. A variety of subtasks such as data generation, choosing the size of the network, training, and validation are required to create a good NN model. For a given modeling problem, the amount of training data or size of the network required is not known in advance. Therefore in a manual NN model development approach, the experience and knowledge of the user are important factors for developing a good model. As well, the manual approach is susceptible to human error and may lead to longer overall model development time. An automated procedure that combines the various NN development tasks, will allow NN models to be developed automatically without a great deal of user intervention. An important motivation of this thesis is the development of an automated modeling technique to develop time domain (TD) NN models. These TD models are then used in both linear and nonlinear circuit applications.

NN models of microwave structures are typically trained with frequency-domain (FD) data from EM solvers. These NN are static models that can be considered as simple input-to-output function mappers. If resonances are present in the FD behavior, the NN model is difficult to train which results in a large network with poor generalization capability. Further training problems arise when the geometrical and/or material properties of the EM structures are considered as variables in the NN model. By using a direct TD formulation, the above issues can be avoided and is a motivation for using the

automated modeling technique to train TD NN models. After training, the TD NN model can be used in EDA tools to represent the EM structures accurately without resorting to expensive EM simulations.

Behavioral modeling of nonlinear circuits is an important area in RF/microwave design. Of particular importance is power amplifier (PA) modeling when modern third-generation (3G) modulation signals are applied. Such signals suffer from nonlinear distortions due to the PA memory effects. TD NN are well suited for learning such memory effects and as a result, the automated modeling technique is used to develop PA behavioral models. These behavioral models are useful for high-level simulations in both TD and FD, and are another motivation for this thesis.

1.2 Thesis Objective

The objective of the thesis is to develop an automated TD modeling technique for linear and microwave circuits using a dynamic NN model called the recurrent neural network (RNN). The RNN is used to develop models for EM structures with variable material/geometrical parameters and PA behavioral models. The significant contributions of this work are as follows:

- For the first time, an automated modeling method using the RNN is proposed [5]. It is an expansion of the existing automatic neural model generation (AMG) algorithm into the dynamic TD modeling area. AMG automatically determines the RNN structure and the training data distribution to achieve good training without user intervention. The automation of RNN training in such a manner allows the overall model development time to be reduced.

- Based on the automated RNN modeling technique, the TD EM behavior of EM structures with variable material and geometry parameters are modeled with the RNN [6]. TD EM simulators are used for training data generation and are automatically driven by AMG. The direct TD formulation is more efficient at handling variable model parameters than FD modeling techniques and can be implemented in conventional circuit simulators.
- The automated TD modeling technique is used in the development of PA behavioral models. The RNN is used to learn the envelope dynamics and the distortion (AM/AM, AM/PM) caused by 3G digital modulation schemes. The resulting PA model is useful as a high-level model to observe the PA behavior in both time and frequency.

1.3 Thesis Organization

The thesis is organized in the following manner.

Chapter 2 is a literature review of NN in RF/microwave modeling and design. Both static NN for EM modeling and dynamic NN for nonlinear circuit modeling are introduced. As well, the AMG algorithm for NN development shall be described.

In Chapter 3, the automated RNN modeling technique is proposed. It is an expansion of the AMG algorithm to support TD model development using the RNN.

In Chapter 4, the automated RNN modeling technique is applied to transient EM modeling of microwave structures for variable material/geometrical parameters. TD EM simulators are used as data generators to train the RNN with the transient responses. The direct TD formulation is more efficient at representing wideband FD behavior for varying

parameters. In addition, a circuit simulator implementation of the RNN is introduced. A couple of examples are shown to demonstrate the method.

Chapter 5 presents the application of automated RNN modeling to behavioral modeling of power amplifiers (PA). The resulting RNN PA model is able to accurately predict both TD distortions and spectral re-growth due to modern third-generation (3G) modulation signals. The RNN PA model is also fast and is therefore useful as a behavioral model for high-level simulation.

Chapter 6 contains the conclusion of the thesis and possible directions for future research.

Chapter 2

Literature Review

2.1 Artificial Neural Networks for RF/Microwave Design

Artificial Neural Networks (NN) have recently emerged as a useful tool in RF/microwave modeling and design [1]. A major reason for the interest over other modeling methods is that NN are capable of modeling any multi-dimensional nonlinear input-output relationship to any desired accuracy. Due to its internal computational simplicity, the NN is as CPU-efficient (fast) as empirical and polynomial or quadratic curve-fitting models but remains capable of representing highly nonlinear behavior. As well, NN are a compact solution requiring a small amount of computer memory resources. This is especially true for multi-dimensional modeling problems when compared with other techniques such as the table look-up method [4].

Before NN models can be used in modern electronic design automation (EDA) or computer-aided design (CAD) tools, the NN must be trained with appropriate data so the correct input-output behavior is “learned”. Typically, measurement and/or simulation data is used to train the NN. Once properly trained, the NN can then generalize the output

behavior accurately for any arbitrary input within the training range. Such a generalization capability makes NN ideal for modeling components or systems where closed-form or analytical formulae are either unavailable or are inaccurate. Also, as new devices and systems are developed and introduced into the RF/microwave marketplace, NN models can quickly be trained for use in EDA tools.

Fast and accurate NN models of frequency-domain (FD) electromagnetic (EM) behavior have been developed to avoid the high computational burden associated with traditional EM solvers. In the literature, NN are used in the EM modeling of bends [7], vias [8], embedded passive components [9], coplanar waveguide (CPW) components [10], spiral inductors [11], VLSI interconnects [12], microstrip circuits [13], patch/slot antennas [14], and bandpass filters [15]. NN models have also been developed for active devices such as MESFET [4, 16, 17], HEMT [18, 19], and HBT [20, 21]. These transistor models can accurately represent the complex semiconductor physics behavior of the devices thereby making them useful within a larger circuit design. As well, entire nonlinear circuit behaviors can be modeled using dynamic time-domain (TD) NN models that contain feedback [22, 23]. Such TD models are useful for behavioral modeling purposes and have been used to model amplifiers, mixers, and even entire receiver systems.

NN models for linear EM modeling purposes can be considered as simple input-output function mappers. Such models are static in nature since the output is only a function of the current NN inputs. The most basic and frequently used NN structure in RF/microwave design is called the multilayer perceptron (MLP). Other common NN are the radial basis function (RBF) and wavelet networks. The various NN share similar

characteristics in that they all contain *neurons* (processing elements) and *synapses* (connections between neurons) but differ in how the information is processed within each neuron. Nonetheless all the NN are suitable as models for EM behavior of microwave structures once proper training is completed. The major issues in NN training are the number of neurons in the network and the training data distribution required to generate a model with accurate generalization. Since these factors are not known in advance for a given modeling problem, a systematic training algorithm called automatic neural model generation (AMG) has been presented [24]. AMG allows for the automatic development of NN models without requiring a lot of user intervention. However even with AMG, the presence of sharp resonances in the EM behavior are difficult to model using only a NN. As well, if the model is to support variable material and geometrical parameters, the training of the EM behavior may be problematic.

Behavioral modeling requires that the TD dynamics of the nonlinear circuits are directly modeled. As a result, static NN are not sufficient and more advanced structures are required that contain state or feedback information. Two major categories of such TD NN are available, namely the dynamic NN (DNN) and the recurrent NN (RNN). These TD NN are trained with TD training data and can represent the nonlinear microwave circuit behavior faster than conventional circuit simulators.

In the next section, the standard MLP shall be described. This will be followed by a review of TD NN based on NN with feedback. Then the AMG algorithm will be reviewed.

2.2 Neural Network Structures

A generic three-layer NN structure is shown in Figure 2.1. The input layer contains neurons that relay the NN inputs to the hidden layer neurons via pathways called synapses. Similarly, the output layer neurons receive the processed input from the hidden layer neurons to calculate the output of the NN. Such an iterative computation from input to output layer is referred to as feedforward and is the distinguishing feature of a class of NN called feedforward NN (FFNN).

Each synapse in the network has some associated weight parameter(s) that, along with the feedforward computation, completely specifies the behavior of the NN. The purpose of NN training is to find the set of weight parameters that best suits a given modeling problem. This usually involves formulating the training process as an optimization problem that minimizes the error between the training data and the NN output. The number of hidden layer neurons is related to the number of synapses and hence optimizable weight parameters within the NN. In general, more hidden neurons are required to model highly nonlinear input-output relationships while fewer neurons can represent simpler problems. However, larger NN structures are more difficult to train and may not have good generalization capability. When training is completed, the final set of parameters encodes all the intelligence of the NN model to represent and generalize the patterns observed in the training data.

Figure 2.1: Three-layer feedforward neural network (FFNN) structure with n inputs, h hidden layer neurons, and m outputs.

2.2.1 Multilayer Perceptron

The most famous FFNN for RF/microwave design is the multilayer perceptron (MLP). It has been proven that any multi-dimensional nonlinear input-output relationship can be modeled to any desired accuracy using a three-layer MLP (MLP3) if enough hidden layer neurons are available [25]. For the MLP3, the information processing in each hidden layer neuron is a two-step procedure. Figure 2.2 describes this entire process graphically for the j^{th} hidden neuron. The inputs are each multiplied with their corresponding synapse weights and added together. The sum (γ_j) is then sent to an activation function ($\sigma(\cdot)$) to generate the hidden neuron output (z_j). The hidden neuron output will then be used in subsequent calculations to produce the MLP3 output.

Mathematically the feedforward for a MLP3 with n inputs, h hidden neurons, and m outputs can be stated as:

For each output y_i ,

$$y_i = \left(\sum_{j=1}^h v_{ij} z_j \right) + v_{i0} \quad (2.1)$$

where the activation function is

$$z_j = \sigma(\gamma_j) \quad (2.2)$$

and from the input layer

$$\gamma_j = \left(\sum_{k=1}^n w_{jk} x_k \right) + w_{j0} \quad (2.3)$$

w_{jk} are the synapse weights between input x_k and j^{th} hidden layer neuron. v_{ij} are the synapse weights between j^{th} hidden layer neuron and output y_i . Note the bias terms (v_{i0} , w_{j0}) are present in the neuron calculations in (2.1) and (2.3).

Figure 2.2: Information processing within the j^{th} hidden layer neuron of the 3 layer MLP (MLP3). The MLP3 has n inputs and m outputs.

The activation function $\sigma(\cdot)$ for the hidden layer neurons is typically a sigmoid function of the form

$$\sigma(\gamma) = \frac{1}{1 + e^{-\gamma}}. \quad (2.4)$$

Other possible hidden layer activation functions are the arctangent

$$\sigma(\gamma) = \frac{2}{\pi} \arctan(\gamma) \quad (2.5)$$

and the hyperbolic tangent function

$$\sigma(\gamma) = \frac{e^{\gamma} - e^{-\gamma}}{e^{\gamma} + e^{-\gamma}}. \quad (2.6)$$

Figure 2.3 shows each of the above activation functions graphically. Theoretically any bounded, continuous, monotonic, and continuously differentiable function is acceptable as an activation function for the hidden layer neurons. The activation functions for the input layer neurons are usually relay functions while the output layer neurons are linear functions as

$$\text{Input relay function} \Rightarrow \sigma(x_k) = x_k \quad (2.7)$$

$$\text{Output linear function} \Rightarrow \sigma(\gamma) = \gamma \quad (2.8)$$

Based on the feedforward computation, the MLP3 can then be totally specified by the number of inputs, hidden neurons, outputs, and the set of synapse weights. Including the contribution of bias terms, the number of weights is

$$\# \text{ weights} = h(n+1) + m(h+1). \quad (2.9)$$

The weights are then organized into a vector called the weight vector, $\mathbf{w}$ defined as

$$\mathbf{w} = \begin{bmatrix} w_{10} \cdots w_{1n} & w_{20} \cdots w_{2n} & \cdots & w_{h0} \cdots w_{hn} & v_{10} \cdots v_{1h} & v_{20} \cdots v_{2h} & \cdots & v_{mh} \end{bmatrix}^T. \quad (2.10)$$

$\in \mathfrak{R}^{h(n+1)+m(h+1)}$

Figure 2.3: Neuron activation functions for hidden layer neurons of the MLP. All the functions are bounded, continuous, monotonic, and continuously differentiable.

The goal of MLP training is to find the weight vector that can represent the training data accurately. This involves solving a multi-dimensional nonlinear optimization problem by minimizing the error between training data and MLP output. However before many gradient-based optimization techniques can be used for such a purpose, the sensitivity of the MLP output to the internal weights is required. The output gradient wrt the internal weight parameters of the MLP can be calculated by differentiating the output and then “back propagating” by chain rule the derivatives through the network. The gradient of the MLP3 with (2.7) and (2.8) starts by first differentiating (2.1) as

$$\frac{\partial y_i}{\partial v_{lj}} = \begin{cases} z_j & \text{if } l = i \\ 0 & \text{otherwise} \end{cases} \quad (2.11)$$

and continuing the back-propagation by

$$\begin{aligned} \frac{\partial y_i}{\partial w_{jk}} &= \frac{\partial y_i}{\partial z_j} \frac{\partial z_j}{\partial \gamma_j} \frac{\partial \gamma_j}{\partial w_{jk}} \\ &= v_j \frac{\partial z_j}{\partial \gamma_j} x_k \end{aligned} \quad (2.12)$$

The middle term in (2.12) is simply the derivative of the activation function (2.2). If the sigmoid function of (2.4) is used, it can be shown that [1]

$$\begin{aligned} \frac{\partial y_i}{\partial w_{jk}} &= v_j \frac{\partial z_j}{\partial \gamma_j} x_k \\ &= v_j z_j (1 - z_j) x_k \end{aligned} \quad (2.13)$$

The output gradient wrt the weights for the MLP3 is formed by combining (2.10), (2.11), and (2.13) into

$$\frac{\partial y_i}{\partial \mathbf{w}} = \left[\frac{\partial y_i}{\partial w_{10}} \dots \frac{\partial y_i}{\partial w_{1n}} \quad \frac{\partial y_i}{\partial w_{20}} \dots \frac{\partial y_i}{\partial w_{hn}} \quad \frac{\partial y_i}{\partial v_{10}} \dots \frac{\partial y_i}{\partial v_{1h}} \quad \frac{\partial y_i}{\partial v_{20}} \dots \frac{\partial y_i}{\partial v_{mh}} \right]^T. \quad (2.14)$$

Another important figure of merit for the MLP3 is the sensitivity of an output wrt the specific input. For a sigmoid activation function the derivative of y_i wrt x_k is simply

$$\begin{aligned}\frac{\partial y_i}{\partial x_k} &= \sum_{j=1}^h \frac{\partial y_i}{\partial z_j} \frac{\partial z_j}{\partial y_j} \frac{\partial y_j}{\partial x_k} \\ &= \sum_{j=1}^h v_{ij} z_j (1 - z_j) w_{jk}\end{aligned}\quad (2.15)$$

With the MLP3 feedforward and back-propagation formulae, the NN training can be performed to find an optimal set of weights to match the training data. For a given training data set L containing P samples ($p=1..P$)

$$L = \{ \mathbf{x}^{(p)} \mid (\mathbf{x}^{(p)}, \hat{\mathbf{y}}^{(p)}) \}, \quad (2.16)$$

the normalized l_2 training error is

$$E(\mathbf{w}) = \frac{1}{2mP} \sum_{p=1}^P \sum_{i=1}^m \left(y_i(\mathbf{x}^{(p)}, \mathbf{w}) - \hat{y}_i^{(p)} \right)^2. \quad (2.17)$$

Differentiating (2.17) wrt the weights gives the following error gradient

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{mP} \sum_{p=1}^P \sum_{i=1}^m \left(y_i(\mathbf{x}^{(p)}, \mathbf{w}) - \hat{y}_i^{(p)} \right) \frac{\partial y_i}{\partial \mathbf{w}} \Big|_{\mathbf{x}=\mathbf{x}^{(p)}}. \quad (2.18)$$

Based on (2.17) and (2.18), the NN training process for the MLP using a gradient-based optimization technique is summarized by the steps in Figure 2.4. Note that since NN training represents a highly multi-dimensional nonlinear problem, there are no guarantees that the training will converge for a given number of training iterations (or epochs). In such cases, user intervention is required to determine how training performance can be improved. As well, only local optimal solutions are usually achievable.

MLP TRAINING ALGORITHM

1. Obtain training data. Initialize MLP weights (w) to small random values. Set $epoch=0$.
2. Compute $E(w_{epoch})$ using (2.17).
3. If $(E(w_{epoch}) < \text{desired accuracy})$ or $(epoch > \text{maximum epoch}) \Rightarrow STOP$ and save w_{epoch} .
4. Compute $dE(w_{epoch})/dw_{epoch}$ using (2.18).
5. Determine a weight update from $E(w_{epoch})$ and $dE(w_{epoch})/dw_{epoch}$ using optimization algorithm.
6. Update weights $\Rightarrow w_{epoch+1} = w_{epoch} + \text{weight update}$. $epoch = epoch + 1$. Goto step 2.

Figure 2.4: Summary of steps to train the MLP. In step 5, many gradient-based optimization algorithms such as back-propagation, steepest descent, conjugate gradient, and quasi-Newton can be used to determine the weight update.

2.2.2 Neural Networks with Feedback

A major area in EDA research is the development of time-domain (TD) models. For a given system identification problem, a variety of techniques can be used to obtain a black-box model based on the input-output signal relationships [26]. The developed model should be a simplification of the original system and capable of characterizing the system behavior over the desired range of model inputs. Since dynamics are present, the TD model will have multiple states or orders. The presence of state or order is an indication that a simple algebraic (static) model is not suitable for TD modeling. In Section 2.2.1, a FFNN called the MLP is described. When the MLP is used for an input-output function mapping, it is a static model that is unable to represent dynamic TD behavior. A simple example can be shown to illustrate how static NN cannot resolve the type of inconsistencies that are associated with dynamic TD behavior.

Consider a single-input single-output (SISO) time series where the input is $u(t) = A\cos(\omega t)$ and the output is $f(t) = A\sin(\omega t)$. A 1-input and 1-output MLP is selected to learn the input-output relationship as shown in Figure 2.5a. Training data sample pairs $(u(t), f(t))$ are then uniformly selected over a single period of the input-output waveforms to train the MLP. The training data distribution is shown in Figure 2.5b. It is clear that as time advances the correct dynamic behavior is represented by a counter-clockwise rotation around the training data space. However, in a static MLP, the dynamic information is not available and for some input $u(t)$ there are two possible valid values for $f(t)$ (in a static sense). For example when $u = \frac{A}{\sqrt{2}}$, f can be either $\frac{A}{\sqrt{2}}$ or $-\frac{A}{\sqrt{2}}$ depending

Figure 2.5: a) 1-input and 1-output static MLP3 to be trained with discrete samples from an input-output TD sequence. b) Training data distribution of the samples of the input-output waveform. A static model will not be able to handle multiple outputs for the same input.

on the current state. This inconsistency highlights the fundamental problem of using static NN to learn TD behavior.

For general TD modeling, a variety of NN solutions have been developed to address this problem. They usually involve incorporating feedback (state) and input memory information to the NN [27]. Two major classes of TD NN, the dynamic NN (DNN) and the recurrent NN (RNN) are now described.

2.2.2.1 Dynamic Neural Network

The dynamic neural network (DNN) has been used to model the behavior of nonlinear circuits such as an amplifier, mixer, and an entire DBS receiver system [23]. It has also been applied to the development of a HEMT model based on large signal TD measurements [28]. It is a continuous-time formulation that is most ideal to describe the nonlinear circuit behavior in modern harmonic balance simulators. The original nonlinear circuit can be described in state-space form as

$$\begin{aligned}\dot{x}(t) &= \varphi(x(t), u(t)) \\ y(t) &= \psi(x(t), u(t))\end{aligned}\tag{2.19}$$

where u and y are the vectors of input and output signals respectively. x are the state variables of the nonlinear circuit and includes nodal voltages, inductor currents, voltage source currents, and charges of nonlinear capacitors. φ and ψ represent nonlinear functions that, for large circuits, could be a large set of nonlinear differential equations. Solving such equations are computationally expensive, and therefore a simpler model that can still represent the TD behavior of the nonlinear circuit accurately is desirable.

The dynamics of (2.19) are reformulated into a n^{th} order differential equation form as

$$y^{(n)}(t) = f(y^{(n-1)}(t), y^{(n-2)}(t), \dots, y(t), u^{(n)}(t), u^{(n-1)}(t), \dots, u(t)). \quad (2.20)$$

The DNN model is derived from (2.20) according to

$$\begin{aligned} y(t) &= v_1(t) \\ \dot{v}_1(t) &= v_2(t) \\ &\vdots \\ \dot{v}_{n-1}(t) &= v_n(t) \\ \dot{v}_n(t) &= f_{NN}(v_n(t), v_{n-1}(t), \dots, v_1(t), u^{(n)}(t), u^{(n-1)}(t), \dots, u(t)) \end{aligned} \quad (2.21)$$

For a SISO system, the DNN can be implemented using the MLP structure when trained with the output (y) and input (u) derivatives as shown in Figure 2.6. The presence of the additional derivative inputs, allows the MLP to learn the TD dynamics by avoiding the inconsistencies as described in Section 2.2.2.

Another useful feature of the DNN is that it is straightforward to implement in nonlinear circuit simulators. The circuit representation of the DNN is shown in Figure 2.7. The state variables (v) are the voltages across unit capacitances while the input (u) is the current through unit inductances.

The DNN can be trained using the input/output harmonic spectrums of the nonlinear circuit. Let $\bar{U}(\omega)$ and $\bar{Y}(\omega)$ be the set of input-output frequency spectrum points of the circuit over a range of frequencies Ω ($\omega \in \Omega$). Let matrix $A(\omega, t)$ be the coefficients of the Inverse Fourier Transform and $A^{(i)}(\omega, t)$ be the i^{th} time derivative of $A(\omega, t)$. The derivative information for the m^{th} circuit output can then be obtained from

$$\bar{y}_m^{(i)}(t) = \sum_{\omega \in \Omega} A^{(i)}(\omega, t) \cdot \bar{Y}_m(\omega) \quad (2.22)$$

and

$$\bar{u}_m^{(i)}(t) = \sum_{\omega \in \Omega} A^{(i)}(\omega, t) \cdot \bar{U}_m(\omega). \quad (2.23)$$

Figure 2.6: DNN model based on the MLP for a single-input, single-output (SISO) system. The derivative information in the input allows for TD modeling (from [23]).

Figure 2.7: Circuit implementation of the DNN model. The state variables (v) are voltages across unit capacitances ($C=1\text{F}$) while the input (u) is the current through unit inductances ($L=1\text{H}$) (from [23]).

Using the relations of (2.22) and (2.23), the initial DNN training can be performed as shown in Figure 2.8. Gradient-based NN training algorithms are typically used to train the DNN structure. Provided that enough hidden neurons and order (n) are present in the DNN structure, the training error will eventually converge to a low value. After successful validation with test data not used in training, the resulting DNN can then be used in nonlinear circuit simulators as a fast, accurate, and compact TD model of nonlinear circuit behavior.

2.2.2.2 Recurrent Neural Network

The recurrent neural network (RNN) has been used to model the behavior of nonlinear circuits such as an amplifier, mixer, and MOSFET [22]. It differs from the DNN in that it is a discrete-time structure that models the finite difference relationship

$$y(k) = g(y(k-1), \dots, y(k-M_y), u(k-1), \dots, u(k-M_u), p) \quad (2.24)$$

where k is the index, M_y is the feedback order, M_u is the input memory, and p is a vector of time-independent parameters. Figure 2.9 shows how the RNN is trained with the input-output TD responses of nonlinear circuits. The presented RNN uses the previous outputs (feedback) plus input history to determine the next output value. Other RNN structures that utilize state information [29], [30] and self-feedback in each neuron [31] have also been shown for nonlinear dynamical systems modeling. Regardless of the specific structure, the recurrent nature of the RNN is a potentially powerful TD model for EDA purposes. However, the discrete nature of the RNN does not lend itself to a convenient circuit simulator implementation and is a major drawback.

Figure 2.8: Training of the DNN using the input and output spectrum of a nonlinear microwave circuit. Once successfully trained, the DNN can be used in circuit simulators as a fast and accurate model of the entire nonlinear circuit (from [23]).

Figure 2.9: Training of the RNN using TD input-output sequences from a nonlinear microwave circuit. Note that the previous outputs (feedback) plus input history is used to determine the next RNN value (from [22]).

Due to the presence of explicit feedback, RNN training is more complex than static NN or DNN training. For an RNN with only output feedback and an internal MLP structure, a few modifications must be made to the output gradient in (2.14) so that the effect of the history is taken into account. The backpropagation through time (BPTT) concept extends regular back-propagation not only through the network but also to previous time instances using the chain-rule [32]. By using the BPTT gradients, gradient-based optimization can be used to perform RNN training. However, such BPTT training is very slow and requires many epochs to converge. As well, if the training waveforms have slowly varying long-term dependencies, the RNN training is difficult because of gradient decay [33]. To improve the robustness of RNN training, second-order training methods based on the Kalman filter have been developed [34]. These training algorithms have better rates of training convergence than BPTT but are computationally more expensive. Recently, work has been published on a simplified Kalman approach that does not use derivative information during RNN training [30].

Another major issue with the RNN is the stability of the structure itself due to the presence of feedback. For some bounded input applied to the RNN, it is possible that the output will eventually blow up and saturate at a very high level. This is caused when the RNN has not been trained appropriately to deal with situations when the output starts to drift. Such instability is of great concern when developing TD models. Currently, the RNN can only be checked according to Lyapunov stability in a post-training step when the internal weight parameters are already set [35]. If the RNN is found to be unstable, the resolution is to re-start training using a different structure or set of initial weights. The idea is to find another local optimum solution to the RNN training problem. However, it

would be ideal and much more time-efficient if the RNN training algorithm itself could maintain global stability while adjusting the internal weights. This is still an open question.

2.3 Automatic Neural Model Generation

Neural model development involves several subtasks like data generation, neural-network selection, training, and validation. In a manual approach, these subtasks are performed in a sequential manner independent of one another. Such an approach requires intensive effort and is prone to human error. As well, the quality of the developed neural model is closely linked to the NN training experience of the designer. Since many designers do not have in-depth knowledge of NN, there is a need for the automation of the NN training process.

RF/microwave modeling problems are often highly nonlinear and multi-dimensional. The number of hidden neurons in a FFNN structure, such as the MLP, to develop an accurate model is not known *a priori*. Too few hidden neurons in the network will result in poor NN training with high error. This phenomenon is called underlearning and is an indication that the network does not possess enough freedom to learn the nonlinearities in the training data. On the other hand, too many hidden neurons may lead to long training times and poor generalization capability. Inaccurate generalization is referred to as overlearning, and represents how the NN has simply memorized the training data but not the patterns between the input and output. Therefore after successful training, the NN should always be validated with data not used in training to check the generalization.

The solution to underlearning is to increase the size of the NN structure. The automatic neural model generation (AMG) algorithm [24], used to train static FFNN models such as the MLP, can automatically adjust the number of hidden neurons during training depending on the underlearning phenomenon. By adaptively growing the NN structure, AMG can select the final structure that is able to achieve the desired training accuracy for a given modeling problem.

A major issue in NN training is the training data distribution. Qualitatively, smooth or linear regions of the model require fewer training data points while nonlinear regions should have more finely sampled points. At the same time, generating too much training data through oversampling could be expensive (e.g., three-dimensional (3-D) EM simulations) and too few samples will lead to the overlearning problem. Therefore AMG provides an intelligent sampling algorithm that attempts to overcome the above problems in a systematic manner. AMG, in combination with automatic data generation (ADG), is able to drive the data generator to produce the training data during the training process. The user only has to specify the input training range of the model, and a neural model can be automatically developed by AMG even when no initial training data is provided. If AMG is successful, the final neural model will not exhibit underlearning and overlearning, but good learning.

When overlearning is noticed after a training stage, more samples from the worst region of the model should be added to the training set and the NN should be trained further. As well, additional validation samples should be obtained to validate the NN in the next training stage. This process can be shown for a two-dimensional

input space in Figure 2.10. The worst region of the NN model is determined by the validation sample that has the highest test error. By continually generating additional training and validation data in the worst sub-region, the model input space is sampled by AMG according to the training quality. Such an approach leads to a more efficient model development process that reduces the overall NN training time. As well, it will lead to a neural model with good learning.

In a stage-wise manner, the AMG algorithm is used to train an n -input m -output NN structure. Figure 2.11 shows the flowchart of the AMG framework to train the NN. Let $\mathcal{R}$ represent a set of regions of the NN input space (n -dimensional $\mathbf{x}$ -space) and E_d represent the user-desired neural model accuracy. Let E_t^k and E_v^k be the training error and validation error respectively for the NN structure (S_k) containing N_h^k hidden neurons at the end of the k^{th} training stage. User defined parameters include the input range of R_0 ($R_0 \in \mathcal{R}$), maximum number of stages (k_{max}), initial number of hidden neurons (N_h^1), underlearning factor (β), and overlearning factor (η). R^* is the subregion containing the validation sample with highest validation error. The training and validation data sets are denoted as L^k and V^k . The number of hidden neurons to add when underlearning is detected is δ .

Note that all the previously independent subtasks are now performed in a single unified process that combines automated NN structure selection and intelligent automatic data generation. As a result, AMG represents an important achievement in the NN development area.

Figure 2.10: a) Training (•) and validation (x) data in a subregion of two-dimensional input space. b) After training, if the validation sample (x) has the highest error among the entire validation data, the sub-region is considered the worst and is then subdivided (star-distribution) to generate new training (P) and validation (Q) samples. Note that the original validation sample is now a training sample in the next stage (⊗) (from [24]).

Figure 2.11: Flowchart of the AMG algorithm to train a NN structure (S) over k stages.

Both automatic data generation and NN training are combined so that a good neural model can be achieved. If some of the training data contains large errors (measurement or accidental errors), Huber quasi-Newton training is performed to ignore the errors (from [24]).

2.4 Conclusions

Artificial neural networks have been introduced as a useful tool for RF/microwave design. The most basic and commonly used NN structure called the MLP has been explained in detail along with the general training procedure. In addition, two NN with feedback have been described for TD modeling purposes. The DNN is shown to be useful to model nonlinear circuit behavior and can easily be incorporated into circuit simulators. The RNN can also represent TD behavior using a discrete-time formulation, but requires more complex training algorithms. As well, the RNN implementation into circuit simulators is not straightforward.

The AMG algorithm has been described for NN training. Currently AMG is only used to train NN models with some static input-output behavior. In the next part of the thesis, AMG will be expanded to facilitate automated RNN training for linear and nonlinear microwave circuit applications.

Chapter 3

Automatic RNN Modeling

3.1 Introduction

The training of feedforward NN (FFNN), such as multilayer perceptron (MLP), involve a variety of subtasks such as data generation, choosing the number of hidden neurons, training, and validation. These tasks, though related, are performed sequentially in an independent manner using a manual NN training framework. Such a methodology is error prone and requires a great deal of user intervention during the NN model development process. For instance, the amount and distribution of the training data to use for producing an accurate NN model with good generalization is not obvious since nonlinear regions require more data points with finer resolution than smooth regions. Another issue is that the number of hidden neurons required for a given modeling problem is not known *a priori*. Too few hidden neurons and the NN is unable to learn the training data (underlearning) while too many hidden neurons may lead to poor generalization capability (overlearning) and long training times. Therefore, the AMG algorithm was developed to combine the various NN development subtasks into one single automated process. Combined with automatic data generation (ADG), AMG can automatically drive a data generator (i.e. EM solver or circuit simulator) to appropriately

create the necessary training data based on the nonlinear and smooth regions of the model. In addition, the number of hidden neurons is automatically adjusted by AMG according to the underlearning condition. Currently, AMG is only able to develop NN models using steady-state information such as FD or DC-bias training data. The resulting NN is a static model where the outputs are only a function of the current inputs.

This chapter proposes an expansion of AMG into TD model development. Due to the recent maturation of TD EM simulators for EM-based design, and the need for behavioral models of nonlinear microwave circuits, AMG is expanded to support the development of dynamic models that are directly trained with TD information. A TD NN structure called the recurrent neural network (RNN) is utilized for such a purpose.

3.2 RNN Macromodel

The RNN macromodel is shown in Figure 3.1 for a time-varying input signal u and output f . The RNN is valid for TD modeling due to the presence of feedback (recurrency) and memory (history). Mathematically let g_{RNN} represent the RNN as

$$f(kT - \tau) = g_{\text{RNN}}(f((k-1)T - \tau), \dots, f((k-M_y)T - \tau), u(kT), u((k-1)T), \dots, u((k-M_u)T), \mathbf{w}, \mathbf{p}) \quad (3.1)$$

where k is the index, T is the step size, M_y is the feedback order, M_u is the input history, $\mathbf{w}$ are the internal weights of the FFNN, and $\mathbf{p}$ is the vector of time-independent parameters. The parameter τ is the delay between the input and output signals.

Nonlinear gradient-based optimization techniques such as back-propagation, conjugate gradient, and quasi-Newton are commonly used to find the set of internal weights ($\mathbf{w}$) that minimize the error between a FFNN and training data. For the RNN, the

Figure 3.1: RNN structure with output feedback (M_y). The RNN is a discrete-time structure trained with sampled input-output data.

presence of feedback adds an additional complexity since the output is not only a function of the current inputs but also the previous outputs. The back-propagation through time (BPTT) concept is used to calculate gradients that include the recurrent nature of the RNN. Using the BPTT gradients, the RNN can then be trained with TD input/output sequences. However to achieve good training for a given set of training waveforms, the RNN must have sufficient feedback order and hidden neurons. In general, many RNN delay steps are required to model transient sequences with non-repeating rapid fluctuations (wrt step size T) while smooth or repeating behavior can be modeled using fewer delays. In addition, the number of hidden neurons should be enough to allow the input to output mapping for every time instance of the RNN. Naturally, many feedback delays and hidden neurons (more weights) result in difficult RNN training that requires a long time to achieve a desired accuracy. The selection of the RNN order is an important issue and will be automated within the AMG process.

3.3 AMG for RNN

Developing a good RNN model requires training with appropriate waveforms and selecting the necessary order and number of hidden neurons. Since the RNN is a dynamic model it is trained with TD output signals (f) to applied input signals (u). If the RNN is unable to learn the training waveform set, the RNN is considered to be in underlearning. The resolution to underlearning is to increase the RNN structure by adding hidden neurons (more freedom) or order (memory) until the training reaches convergence. Once RNN training is completed, the model must be verified with validation waveforms that have similar properties to the training set. For instance, if the RNN has been trained with

waveforms covering a certain bandwidth range, the validation waveforms should also lie within the bandwidth range but not be the same as the training set. If the validation error is low, the RNN is said to have achieved good learning in the bandwidth of interest, otherwise the RNN suffers from a condition called overlearning. Overlearning indicates that the RNN simply memorized the training data and that it is unable to generalize accurately. Steps to alleviate overlearning are to add more waveforms to the training set and continue training the RNN structure. Upon successful training, the RNN must be validated again with different waveforms.

AMG automates the RNN training process by increasing the structure size when underlearning is detected. As well, AMG can be used to drive a data generator to obtain additional training waveforms when overlearning is detected. AMG is also useful for reducing the size of the final RNN structure while maintaining good learning.

Automatic data generation (ADG) of the training waveforms is possible if the modeling problem involves some time-independent parameters (p) such as material or geometrical properties. AMG can automatically sample these parameters and drive the data generator during training to obtain the training waveform set. During data generation, the same input excitation (u) is utilized so the differences in the various training waveforms are a result of the changes in the parameters and not because of changes in the input excitation. To facilitate the use of AMG, a new formulation of the RNN called the RNN-trainer is introduced. Figure 3.2 shows the RNN-trainer structure and its relationship to a conventional RNN. The RNN-trainer re-maps the input signal to an input index variable that represents the current time (context) of the RNN. For a given parameter, the evaluation of the RNN-trainer structure involves sweeping the index

Figure 3.2: Relationship between RNN and RNN-trainer. The RNN-trainer structure uses only the parameters (p) to generate the entire output training waveform by sweeping the index (k) from 0 to N_t-1 (# samples = N_t).

variable incrementally. Note that training the RNN-trainer until good learning is achieved will result in a set of internal weights and structure that can then be used in the original RNN formulation. The purpose of RNN-trainer is to allow a dynamic model such as the RNN to be trained using AMG.

Let L and V be the sets of training and validation samples for the RNN-trainer respectively as

$$L = \{p_i | (p_i, k, \hat{f}_i)\} \quad (3.2)$$

and

$$V = \{p_j | (p_j, k, \hat{f}_j)\}. \quad (3.3)$$

Assuming that each waveform is represented by N_t uniformly spaced samples, the RNN-trainer structure is proposed for each sample as

$$f(k) = f(p, k) = \begin{cases} g_{\text{RNN}}(u(0), w, p) & , k = 0 \\ g_{\text{RNN}}(f(0), u(T), u(0), w, p) & , k = 1 \\ g_{\text{RNN}}(f(T), f(0), u(2T), u(T), u(0), w, p) & , k = 2 \\ \vdots & \\ g_{\text{RNN}}(f((N_t - 2)T), \dots, f((N_t - M_y)T), u((N_t - 1)T), \\ \quad u((N_t - 2)T), \dots, u((N_t - M_u)T), w, p) & , k = N_t - 1 \end{cases} \quad (3.4)$$

The l_2 error of RNN-trainer for a single waveform is calculated by sweeping the index incrementally as

$$\begin{aligned} e(p) &= \left[\frac{1}{2} \sum_{k=0}^{N_t-1} |f(p, k) - \hat{f}(k)|^2 \right]^{1/2} \\ &= \left[\frac{1}{2} \sum_{k=0}^{N_t-1} |g_{\text{RNN}}(\dots, u(k), \dots, w, p) - \hat{f}(k)|^2 \right]^{1/2} \end{aligned} \quad (3.5)$$

The normalized l_2 training error is

$$E_l = \frac{1}{N_l} \sum_{p_i} (e(p_i))^2 \quad (3.6)$$

where N_l is the number of waveforms in the training set L . Similarly, the normalized l_2 validation error is

$$E_v = \frac{1}{N_v} \sum_{p_j} (e(p_j))^2 \quad (3.7)$$

where N_v is the number of validation waveforms in V . To train the RNN-trainer structure, the error gradient wrt to the internal weights is required by the optimizer. One of the components of the error gradient is the RNN-trainer BPTT gradient, which for each training sample is calculated using

$$\begin{aligned} \frac{df(k)}{dw} &= \frac{d}{dw} f(p, k) \\ &= \begin{cases} \frac{\partial}{\partial w} [g_{\text{RNN}}(u(0), w, p)] & , k = 0 \\ \frac{\partial}{\partial w} [g_{\text{RNN}}(f(0), u(T), u(0), w, p)] + \frac{\partial f(T)}{\partial f(0)} \cdot \frac{df(0)}{dw} & , k = 1 \\ \frac{\partial}{\partial w} [g_{\text{RNN}}(f(T), f(0), u(2T), u(T), u(0), w, p)] \\ \quad + \frac{\partial f(2T)}{\partial f(T)} \cdot \frac{df(T)}{dw} + \frac{\partial f(2T)}{\partial f(0)} \cdot \frac{df(0)}{dw} & , k = 2 \\ \vdots \\ \frac{df(jT)}{dw} = \frac{\partial}{\partial w} \left[g_{\text{RNN}}(f((j-1)T), \dots, f((j-M_y)T), \right. \\ \quad \left. u(jT), u((j-1)T), \dots, u((j-M_u)T), w, p) \right] \\ \quad + \sum_{m=1}^{M_y} \frac{\partial f(jT)}{\partial f((j-m)T)} \cdot \frac{df((j-m)T)}{dw} & , k = j \end{cases} \end{aligned} \quad (3.8)$$

By combining BPTT into AMG using (3.8), the effect of feedback in the RNN has been taken into account and the training can proceed using standard gradient-based algorithms. Depending on the neuron activation function and the FFNN structure used

within the RNN, a variety of back propagation formulae for the individual chain-rule derivative terms in (3.8) are available in the literature [1].

Given a user-defined model accuracy threshold E_d , RNN-trainer underlearning is detected by AMG when the training error (E_l) remains roughly the same for many consecutive stages and is higher than the threshold. As well, if the training error increases dramatically after a stage, it is an indication that the order of the structure should be increased to learn the TD dynamics.

Similarly, AMG can detect overlearning once the training error converges below the accuracy threshold, by comparing the validation error and threshold. If the validation error is much larger than the desired accuracy threshold, overlearning has been detected. Table 3-I summarizes how AMG detects underlearning and overlearning after the n^{th} training stage.

TABLE 3-I: AMG DETECTION OF RNN-TRAINER UNDERLEARNING AND OVERLEARNING AFTER N^{TH} TRAINING STAGE

<i>Condition</i>	<i>Check</i>	<i>Symptom</i>	<i>AMG Resolution</i>
$E_l^n \approx E_l^{n-1}$ for many stages	$E_l^n > E_d$	Underlearning	Add neurons/dynamic order. Re-train with existing training waveform set.
$E_l^n \gg E_l^{n-1}$	“	“	“
$E_l^n < E_d$	$E_v \gg E_d$	Overlearning	Activate data generation. Restart training of existing structure with expanded training waveform set.

AMG automatically determines the region of the time-independent parameter space (p -space) where to generate additional training and validation waveforms when overlearning is detected. This differs from conventional AMG for static NN training, where the entire NN input space is used in the training data distribution. The p -space is the static sub-region of the total input space (x -space) of the FFNN within the RNN-trainer structure. The other FFNN inputs associated with the input signal u and output feedback represent the dynamic sub-region of the x -space. The training distribution in the dynamic sub-region is set by the trajectories of the RNN-trainer due to the input u and output feedback. If u is a large bandwidth signal, the dynamic sub-region will be well covered by the trajectory information and the resulting training will lead to a more robust RNN model that can accurately represent the output behavior to a wider range of input signals (with similar or less bandwidth and statistical properties). However, since the same input u is used to generate waveforms for the entire p -space, the AMG training will focus primarily on learning the effect on a single dynamic waveform trajectory due to various p .

Initially, AMG samples the p -space in a star distribution according to the input range for p specified by the user and starts the training process for the RNN-trainer structure. When data generation is activated due to overlearning, the validation sample with the largest error

$$p^* = \arg \max e(p_j) \quad (3.9)$$

is used to determine the region of the p -space that produces a dynamic effect not seen in the training waveforms. AMG selects finer grid samples within the smaller sub-space about p^* (p^* -space) and drives the data generator to obtain the respective TD responses.

The newly generated waveforms are then divided and added to the training set L and validation set V . AMG re-starts training the RNN-trainer with the expanded training waveform set in an attempt to solve the overlearning. The entire iterative process is described by the flowchart in Figure 3.3. Through the continual searching of the validation waveform with highest error, AMG is able to build the necessary training and validation waveforms to achieve good generalization in an intelligent manner that avoids inefficiencies such as the oversampling of the input parameter space and the generation of too much training data. As well, since a grid sampling concept is used throughout, the p -space is well covered.

Once good learning is achieved, AMG can also attempt to reduce the RNN-trainer structure to create a more compact model. Figure 3.4 shows how AMG attempts to create a more optimal structure.

3.4 Summary

The combination of automatic structure selection in the RNN-trainer framework and data generation allows AMG to develop a good RNN model in a systematic manner that does not require a great deal of user intervention. The described automatic RNN modeling technique is compared to the conventional AMG algorithm in Table 3-II. It is used in the subsequent chapters for both linear and nonlinear microwave circuit applications.

Figure 3.3: Flowchart showing the process to achieve good training of a RNN model.

AMG automates the process by using the RNN-trainer structure.

Figure 3.4: Flowchart showing how AMG attempts to reduce the final RNN-trainer structure to a more compact model.

TABLE 3-II: COMPARISON BETWEEN AUTOMATIC RNN MODELING AND AMG

	<i>Model Type</i>	<i>Training Data Format</i>	<i>Adjustable Parameters During Training</i>
AMG	Static	$\{x_i \mid (x_i, \hat{y}_i)\}$	# hidden neurons
Automatic RNN Modeling	Time-domain	$\{p_i \mid (p_i, k, \hat{f}_i)\}$ $p_i \in x_i$	1. #hidden neurons 2. dynamic order

Chapter 4

Transient EM Modeling Using RNN

4.1 Introduction

TD EM modeling has recently become important due to the maturation of solvers based on algorithms such as transmission line matrix (TLM) [36]. TD EM solvers are efficient in obtaining wideband information of microwave structures in a single transient simulation. These tools solve the field equations by performing a mesh analysis of the entire space within and around the EM structure. Depending on the size of the geometry and resolution of the mesh, the computational expense in obtaining the transient EM responses can be high. To develop fast macromodels of microwave structures for high-level simulation and optimization, only the boundary TD EM behavior is important and will be modeled.

Transient EM phenomena for microwave structures are described by Maxwell's equations, which are a set of coupled linear partial differential equations (PDE) relating electric fields (E) and magnetic fields (H) [37]. As a result, the transient EM response at a given location and time in the structure is not simply an algebraic function of an input elsewhere. The precise PDE relating the response to the input at a given time instance can be converted via discretization into a finite difference equation involving multiple time

points (history). RNNs are ideal for modeling such finite difference relationships and therefore are capable of learning the discrete-time EM responses at the boundaries of a given structure.

Indirect approaches to linear TD modeling such as state space equations (SSE) [9], [38], [39] or pole/residues [40], [41] are prevalent in the literature. These macromodels are based on using the FD concept of poles from an EM structure response to generate equivalent circuits [42], [43] for circuit simulator implementation. As the material or geometrical parameters are changed, these poles move in trajectories containing discontinuities due to breakaway points [44]. These discontinuities result in many non-contiguous patterns of poles that are not easy to characterize when the geometry or material parameters are considered as variables to the model. The direct TD formulation with RNN can more efficiently handle such cases of variable geometrical or material model parameters.

4.2 RNN Training with Transient EM Data

Figure 4.1 shows how the EM structure should be set up in an EM simulation to obtain the necessary responses for RNN training. For 2-port passive structures, only three sets of port responses are required for RNN training. An input excitation waveform, $u_{inc}(t)$ is applied to Port 1. The excitation should be capable of establishing a dominant mode of propagation within the structure and can represent either voltage, E-field, or H-field as allowed by the TD EM solver. The resulting port responses $f_1(t)$ and $f_{21}(t)$ will be used to train RNN_1 and RNN_{21} respectively. The same excitation is then applied to Port 2 and the port response $f_2(t)$ is used for RNN_2 training. All three sets of port responses

Figure 4.1: EM simulation setup for EM data generation.

should be consistent with the input excitation and represent the dominant behavior of the EM structure. The TD simulations are then repeated for different combinations of geometrical and material parameters to build the entire data sets for RNN training.

A variety of excitation waveforms can be used for generating RNN training data. Table 4-I lists some of the major categories of waveforms and the impact on training.

TABLE 4-I: TRANSIENT EXCITATION WAVEFORMS FOR GENERATING RNN TRAINING DATA

$u_{inc}(t)$ waveform	Bandwidth (theoretical)	RNN Order	RNN Training
Impulse	Infinite	Large M_y	Difficult
Gaussian (variance= σ^2)	$BW \propto 1/\sigma$	Medium M_y	Moderate
Sinusoid (frequency = ω)	Single frequency	Small M_y	Easy

The excitation waveform should have sufficient excitation (bandwidth) to produce port responses to train the model. Though the impulse response completely characterizes the system behavior for all frequencies, it is very challenging to train since it is usually not smooth and contains many rapid fluctuations. The sinusoidal response is simple to train but does not have sufficient excitation for the RNN macromodel. A Gaussian pulse is a suitable candidate due to the bandwidth and the RNN training capability.

The $f_{21}(t)$ response may contain a relatively long initial output delay (τ) due to the time it takes for the EM power to propagate from one port to another. Direct RNN training of such a delayed waveform may require a large RNN order and input history,

which will slow down training and compromise accuracy. The initial delay is removed from training by setting the training data ($f_{2I}(kT)$) as the time-advanced version of the simulation output ($f_{2I}(t)$) using

$$f_{2I}(kT) = f_{2I}(kT + \tau). \quad (4.1)$$

The TD EM simulation should run until all the transient responses to the excitation input decay to zero. As a result, the port responses can be quite long with many samples, which will cause an additional slow down of RNN training. A simple heuristic is to reduce the length of the training sequences by considering fewer time samples in accordance with the Nyquist interval. If the input excitation signal has bandwidth, BW and a simulation time step of T_{EM} , the RNN sampling interval T in (3.1) can be set according to

$$T_{EM} < T < \text{Nyquist Interval} = \left(\frac{1}{2(BW)} \right). \quad (4.2)$$

Sampling the EM data anywhere in the interval in (4.2) prevents aliasing problems while reducing the length of the training sequences. However the final choice of sampling interval should be the maximum value possible before significant sampling distortion of the EM responses occurs.

4.3 Circuit Simulator Implementation of RNN

Macromodel

After successful training, RNN_1 , RNN_{21} , and RNN_2 are combined into a 2-port sub-circuit component for implementation in circuit simulators. The Fourier transforms (**F**) of the input excitation and RNN port responses are first calculated as

$$\begin{aligned}
U(\omega) &= \mathbf{F}(u_{inc}(k)) \\
F_{RNN,1}(\omega) &= \mathbf{F}(f_{RNN,1}(k)) = \mathbf{F}(g_{RNN,1}(\dots, u_{inc}(k), \dots, \mathbf{p})) \\
F_{RNN,21}(\omega) &= \mathbf{F}(f_{RNN,21}(k)) = \mathbf{F}(g_{RNN,21}(\dots, u_{inc}(k), \dots, \mathbf{p})) \\
F_{RNN,2}(\omega) &= \mathbf{F}(f_{RNN,2}(k)) = \mathbf{F}(g_{RNN,2}(\dots, u_{inc}(k), \dots, \mathbf{p}))
\end{aligned} \tag{4.3}$$

$U(\omega)$ is the Fourier transform of the input excitation while $F_{RNN,1}(\omega)$, $F_{RNN,21}(\omega)$, and $F_{RNN,2}(\omega)$ are the spectrums of the RNN output for a given set of independent parameters ($\mathbf{p}$) and input excitation. Note that these spectrums represent the behavior of the system for varying parameters without the need to estimate any poles. If RNN training is good, accurate spectrums can be generalized for any parameters within the training range. As well, any resonances present in the spectrum are modeled using a pure NN method without resorting to knowledge-based techniques such as external resonant circuits [11].

Using the RNN spectrums from (4.3), the 2-port S-parameters of the RNN macromodel can then be calculated using [45]

$$S_{11}(\omega) = \frac{F_{RNN,1}(\omega) - U(\omega)}{U(\omega)} \tag{4.4}$$

$$S_{21}(\omega) = \frac{F_{RNN,21}(\omega)e^{-j\omega\tau}}{U(\omega)} \sqrt{\frac{Z_{01}}{Z_{02}}} = S_{12}(\omega) \tag{4.5}$$

$$S_{22}(\omega) = \frac{F_{RNN,2}(\omega) - U(\omega)}{U(\omega)} \tag{4.6}$$

where Z_{01} is the characteristic impedance of Port 1 and Z_{02} is the characteristic impedance of Port 2. Note that the output delay τ previously removed from RNN_{21} in (4.1) is added as a phase shift in (4.5). The 2-port S-parameter values can then be converted to other parameters as needed [46]. Assuming that the original training data sequences are passive, and the RNN are trained to a very high accuracy, the extracted parameters should mostly exhibit passive behavior over the bandwidth of the RNN

macromodel (i.e. only absorb real power at all frequencies). For frequencies where the extracted parameters are not passive due to numerical error or inaccurate training, a slight correction is needed to enforce the passivity before applying to the overall circuit matrix for analysis. Optimal approaches to passivity enforcement are also available [47] but have not yet been applied to NN modeling purposes.

4.4 WR-28 Waveguide Example

The RNN macromodel is demonstrated using a WR-28 rectangular waveguide example from a TD EM solver called *MEFiSTo* [48]. The top-view of the waveguide geometry with full height conducting posts is shown in Figure 4.2. WR-28 waveguides have a TE_{10} mode of propagation in the Ka-band (26.5-40Ghz). The pass band characteristics of the waveguide are controlled by the location of the conducting posts.

An input Gaussian excitation pulse with an approximate bandwidth of 40Ghz ($\sigma \approx 6.63$ ps) is launched as a TE_{10} wavefront propagating in the x-direction for a 3000 time step simulation ($T_{EM} = 1.5249762$ ps/step) till all the transient responses decay to zero. The input ($u_{inc}(t)$) and resulting port responses ($f_1(t)$, $f_{21}(t)$, $f_2(t)$) of the potential in the z-direction (V_z) are collected for RNN training.

A delay of 75 samples ($\tau = 75T_{EM}$) is removed from $f_{21}(t)$ using (4.1). In addition, the training sequences are shortened by re-sampling with $T = 4T_{EM} = 6.1$ ps which is about half of the Nyquist interval of 12.5ps.

Figure 4.2: Top view of WR-28 waveguide with dimensions d between conducting posts.

Three geometries are used in RNN training. Since the overall geometry remains symmetric for all d , RNN_1 and RNN_2 represent the same dynamics and a single RNN structure can be used to represent both behaviors. The automatic RNN modeling technique achieves a final structure for RNN_1 with $M_y=M_u=12$ and 20 hidden neurons after approx. 5 hours of training time (including data generation). Furthermore, the final RNN_{21} has $M_y=M_u=16$ and 20 hidden neurons. The average l_2 training error for RNN_1 is 0.087% while RNN_{21} is 0.649%. Parts of the TD port responses are shown in Figure 4.3. The TLM calculation of the transient port responses takes approximately 4.2 seconds while the RNN macromodel requires only 3.6 seconds. For practical examples requiring much longer EM simulation times, the RNN macromodel speed benefit would be more pronounced. The 2-port behavior of the RNN macromodel is shown in Figure 4.4.

4.5 Microstrip Filter Example

The next example is a microstrip filter also from *MEFiSTo*. The top view of the 2-port structure is shown in Figure 4.5 with a user-defined dimension, L . It is desired to model the 2-port behavior of the filter over a bandwidth of 4.5 Ghz for L between 5mm and 19mm. For modeling purposes, the microstrip line is approximated as a purely transverse EM (TEM) line where the E-field is perpendicular to the wave propagation direction. Therefore a TEM excitation waveform injected into the filter in the x-direction has an E-field in the z-direction. An input TEM Gaussian pulse with an approximate bandwidth of 4.5Ghz ($\sigma \approx 58.9\text{ps}$) is launched for a simulation of 4097 time steps ($T_{EM} = 1.66782 \text{ ps/step}$) until all the port responses decay to zero. The resulting E-field in the z-direction (E_z) at the ports is used for RNN training.

Figure 4.3: Comparison between waveguide RNN responses (-) and TLM responses (■) for various d . In the f_{21} response, an initial output delay has been removed before training.

Figure 4.4: 2-port frequency responses of RNN sub-circuit for various d of the WR-28 waveguide example.

Figure 4.5: Microstrip filter with dimension L .

An output delay of 293 samples ($\tau=293T_{EM}$) in $f_{21}(t)$ is removed before RNN training. The Nyquist interval is calculated as 111.1ps, so selecting a sampling interval of $T=25T_{EM}=41.7\text{ps}$ leads to port responses that are shortened without adding a significant amount of sampling distortion.

The automated RNN technique is used to train all three RNN to represent the port dynamics of the filter. The first RNN is trained from a small initial starting structure and takes approx. 11 hours (including data generation) to achieve good learning. The other two RNN are trained by starting with a structure with the same order and number of hidden neurons as the previously converged RNN. This allows for a speed up in training convergence. Table 4-II shows the final RNN structures with good training results.

TABLE 4-II: RNN TRAINING RESULTS FOR MICROSTRIP FILTER EXAMPLE

	# of Hidden Neurons	Final RNN Order ($M_y=M_u$)	Average l_2 Error (%) (15 geometries)
RNN ₁	20	17	0.272
RNN ₂₁	20	17	0.440
RNN ₂	20	17	0.281

The transient port responses are shown in Figure 4.6 for three geometries. The transient port responses of 15 filter geometries using TLM requires approximately 39 seconds while the RNN macromodel takes about 10 seconds. For more complex EM structures, the speed up using the RNN macromodel will become even more pronounced. The circuit simulation results for the 2-port RNN sub-circuit component are shown in Figure 4.7.

Figure 4.6: Comparison between microstrip RNN responses (-) and TLM responses (■) for different L .

Figure 4.7: Frequency responses of 2-port RNN sub-circuit for $L=12$ mm, $L=14$ mm, $L=16$ mm of the microstrip example.

Chapter 5

RNN Behavioral Modeling of Power

Amplifiers

5.1 Introduction

This chapter presents application of RNN to model nonlinear microwave circuits. Specifically, the automatic RNN technique is used to develop high-level behavior models of power amplifier (PA). PA behavioral modeling typically involves characterizing the input to output amplifier signal relationship using a black-box approach based on numerical algorithms such as polynomial/analytical functions or Volterra series [49]. These models are either memoryless, quasi-memoryless, or with memory. To accurately model dynamic PA distortion effects such as AM/AM and AM/PM, system memory must be considered. Accurate modeling of AM/PM and AM/PM of PAs is important in modern digital communication systems since these distortions lead to a deterioration of the overall signal-to-noise ratio. To specifically model such distortions for wideband digital modulation schemes such as W-CDMA and CDMA 2000, only the input-output signal envelopes are required. As a result, the developed models are only useful in the pass-band centered about the RF frequency.

5.2 Power Amplifier Envelope Model

Figure 5.1 shows a PA envelope model with modulated input and transmitted signal.

The nonlinear relationship of the PA can be expressed as

$$\begin{aligned}
 y(t) &= I_{out}(t) + jQ_{out}(t) \\
 &= K(x(t)) \\
 &= K(I_{in}(t) + jQ_{in}(t)) \\
 &= K_1(I_{in}(t), Q_{in}(t)) + jK_2(I_{in}(t), Q_{in}(t)).
 \end{aligned} \tag{5.1}$$

K is a nonlinear complex function with memory representing the PA behavior. K_1 and K_2 are the sub-functions between the applied input In-phase (I_{in}) and Quadrature-phase (Q_{in}) signals and I_{out} and Q_{out} respectively. The dynamic AM/AM and AM/PM distortions of the PA are indicated by [50]

$$\text{AM/AM} = \frac{\|y(t)\|}{\|x(t)\|} = \sqrt{\frac{I_{out}(t)^2 + Q_{out}(t)^2}{I_{in}(t)^2 + Q_{in}(t)^2}} \tag{5.2}$$

and

$$\begin{aligned}
 \text{AM/PM} &= \angle y(t) - \angle x(t) \\
 &= \text{unwrap} \left(\tan^{-1} \left(\frac{Q_{out}(t)}{I_{out}(t)} \right) \right) - \text{unwrap} \left(\tan^{-1} \left(\frac{Q_{in}(t)}{I_{in}(t)} \right) \right)
 \end{aligned} \tag{5.3}$$

From (5.2) and (5.3) the time-varying nature of the AM/AM and AM/PM distortions are clear. Various NN methods to model these distortions have been proposed. For instance, the input history of $I_{in}(t)$ and $Q_{in}(t)$ has been used to train time-delay NN (TDNN) to learn the K_1 and K_2 relations from (5.1) [51]. As well, an approximate technique to directly learn the distortions from the input envelope ($\|x(t)\|$) has been shown using time-delay radial basis function networks (RBF) [52]. However the RNN

Figure 5.1: PA envelope behavioral model for input $x(t)$ and transmitted signal $y(t)$.

will allow a more complete and compact representation of the distortions due to the presence of feedback.

The complex envelope model of the PA is represented by two RNNs as in Figure 5.2. The RNN training data is obtained by applying a digitally modulated signal with a certain channel bandwidth (centered about the RF frequency) to the PA. The channel bandwidth of the RNN training data indicates the range of the envelope dynamics to be modeled. By selecting a modulation scheme with wide bandwidth, such as 3G WCDMA, the PA envelope model will generalize well for other more narrowband modulation schemes with similar statistics. The automatic RNN modeling process is ideal for finding the suitable RNN structures for the envelope PA model. Once good training is achieved, the envelope model can be used to accurately investigate the effect of AM/AM and AM/PM distortions under various modulations and signal waveforms. As well, the spectral regrowth of the PA can be observed. The entire RNN modeling procedure is demonstrated with the following example.

5.3 RFIC Power Amplifier Example

The RFIC PA in *Agilent-ADS* [53] is used to demonstrate the use of RNN for modeling AM/AM and AM/PM distortion. The training data is generated using a 3G WCDMA input signal with average power (P_{av}) of 1 dBm and center frequency of 980 MHz. The channel bandwidth (chip rate) is 3.84 MHz. The In-phase RNN (K_1) and Quadrature-phase RNN (K_2) are each trained by AMG using 1025 simulated input-output samples representing 256 symbols. A user-defined RNN with $M_y=M_u=5$ and 5 hidden neurons is selected as a starting point for the AMG process. In a step-wise manner, AMG

Figure 5.2: PA envelope behavioral model using RNN. Each RNN learns the nonlinear functions, K_1 and K_2 from (5.1).

trains the structure and increases or decreases the RNN order during training. Table 5-I shows the final trained RNN results. Figure 5.3 shows a part of the training waveforms.

TABLE 5-I: RNN TRAINING RESULTS FOR RFIC PA EXAMPLE

RNN	# of Hidden Neurons	Final RNN Order ($M_y=M_u$)	l_2 Error (%)
K ₁	5	1	0.8785
K ₂	5	1	0.8787

Both of the RNN are validated with envelope data not used in training. A $\pi/4$ DQPSK modulated pseudo-random binary signal (NADC) is applied to the RFIC amplifier and RNN envelope model. The channel bandwidth is only 24.3 kHz and is within the training data range. The RNN are also verified with CDMA-2000 type modulation of bandwidth 1.2288 MHz. Figure 5.4 shows some validation waveforms to highlight that the RNN is able to generalize accurately the PA envelope behavior even though the validation data has a different sampling period and time window than the training data. Table 5-II summarizes the validation results. The accurate validation indicates that AMG was able to achieve good training of the RNN for this example.

TABLE 5-II: RNN VALIDATION

Modulated Input ($P_{av}=0$ dBm, $f_c=980$ MHz)	Channel Bandwidth	Sample Period	RNN Test Error (%) (l_2 norm)	
			K ₁	K ₂
NADC	24.3 KHz	4.12 μ s	0.4611	0.4424
CDMA2000	1.2288 MHz	0.203 μ s	0.7304	0.643
3G WCDMA (not training)	3.84 MHz	65.1 ns	0.3635	0.4206

a)

b)

Figure 5.3: RNN training results. (a) $I_{out}(t)$ comparison between *Agilent-ADS* and K_1 RNN. (b) $Q_{out}(t)$ comparison between *Agilent-ADS* and K_2 RNN.

a)

b)

Figure 5.4: RNN validation. (a) $I_{out}(t)$ comparison between *Agilent-ADS* and K_1 RNN for NADC signal. (b) $Q_{out}(t)$ comparison between *Agilent-ADS* and K_2 RNN for CDMA-2000 signal.

Once good learning has been achieved, the AM/AM distortion can be extracted using (5.2). The AM/AM is shown in Figure 5.5 as a function of the instantaneous input power of the 3G WCDMA signal used in training. Similarly, the AM/PM distortion can be modeled with the RNN by using the envelope information in (5.3). Figure 5.6 shows the comparison between simulated results and the RNN for the AM/PM distortion.

The spectral re-growth due to the PA can also be observed using the RNN PA model. This is an important benchmark of the PA to determine if it is dumping too much power into adjacent channels during transmission. Figure 5.7 shows the spectral re-growth around the channel.

The RNN envelope model can accurately capture the dynamic AM/AM and AM/PM distortions of the RFIC PA example. Table 5-III is presented as a comparison between various structures that can be used to represent the In-phase output ($I_{out}(t)$) of the RFIC PA example. Note that the use of feedback in the RNN leads to a more compact model (lower order) for fewer hidden neurons while maintaining good validation behavior. As well, by using AMG, the most compact RNN structure to model the PA envelope response is found automatically.

TABLE 5-III: RFIC PA MODEL COMPARISON FOR IN-PHASE (K_1) RELATIONSHIP

Model Type	# hidden neurons	Structure $I_{in}(t) \rightarrow M_{u1}$ delays $Q_{in}(t) \rightarrow M_{u2}$ delays	l_2 Training Error (%)
TDNN ($M_y=0$)	3	$M_{u1} = M_{u2} = 9$	0.9916
	4	$M_{u1} = M_{u2} = 6$	0.9139
	5	$M_{u1} = M_{u2} = 1$	0.89
RNN	3	$M_y = M_{u1} = M_{u2} = 3$	1.1252
	4	$M_y = M_{u1} = M_{u2} = 2$	1.011
	5	$M_y = M_{u1} = M_{u2} = 1$	0.8785

Figure 5.5: AM/AM distortion between simulation and RNN PA behavioral model for 3G WCDMA training sequence. Note the gain variation due to the PA memory effects. (The low P_{in} point can be better matched with additional training at low power).

Figure 5.6: AM/PM distortion between simulation and RNN PA behavioral model for 3G WCDMA training sequence. This nonlinear distortion is important to model because of the impact on phase-shift type modulation schemes.

Figure 5.7: Spectral re-growth of RFIC PA for the 3G WCDMA training sequence (chip rate = 3.84 MHz). The RNN PA model accurately matches the circuit simulation results.

Through training with the I-Q waveforms of digitally modulated signals, the RNN is capable of learning the dynamics of the PA that can be used to observe the AM/AM, AM/PM distortions, and spectral re-growth. Other PA model formulations that attempt to directly model the AM/AM and AM/PM using magnitude and phase information are also available [52]. However, RNN training in such formulations is difficult since the phase behavior must be directly learned. Another issue is that the phase usually varies very widely for different modulation schemes, so the resulting model will not have good generalization capability.

A final major benefit of using RNN as a PA behavioral model is the improved computational speed over conventional circuit simulators. For the RFIC PA example, *Agilent-ADS* requires approximately 100 seconds to run the entire envelope simulation for the 3G WCDMA input used to generate the training sequences. As a comparison, each RNN reproduces the accurate output for same 3G WCDMA input in only 0.16 seconds. Clearly the RNN is a fast, accurate, and compact model for PA modeling purposes. Therefore, the automatic RNN modeling for PA behavioral modeling is an important application.

Chapter 6

Conclusions and Future Research

6.1 Conclusions

The automatic RNN modeling technique, based on the AMG algorithm, has been used to create TD models for both linear and nonlinear microwave circuit behaviors. The automated technique reduces the manual effort required by the user during RNN training, which leads to a shorter overall model development time. AMG is used in RNN training so that the order can be automatically selected based on the error criterion. As well, AMG can generate additional training data waveforms by automatically driving the data generator when needed.

For linear EM modeling, a TD EM solver can be driven in the appropriate manner so that the training waveforms are sufficient to develop a RNN with good generalization for various material and geometrical parameters. The developed RNN models are faster and as accurate as EM simulations and are useful for repeated analyses such as optimization. As well, the direct TD formulation is more efficient in modeling variable material/geometrical parameters than FD approaches. The RNN macromodel is implemented into a circuit simulator as a single circuit component suitable for larger

level simulation. A WR-28 waveguide and microstrip filter model have been demonstrated using the automatic RNN modeling method.

Automatic RNN modeling has also been applied to model nonlinear power amplifier (PA) behavior. An envelope formulation is used to specifically learn the AM/AM and AM/PM distortions due to digital modulation signals such as 3G WCDMA. The automatic RNN modeling technique is able to select the necessary order during training to learn these TD distortions caused by the PA memory effects. The RNN PA model is then able to accurately model the amplifier behavior in both time (AM/AM, AM/PM distortions) and frequency (spectral re-growth). The PA model also shows good generalization for other modulation schemes with narrower bandwidth and similar statistical properties. As a result, it is useful as a high level PA behavioral model.

This research work has shown the application of automated NN modeling for TD applications. It represents further EDA research within the RF/microwave design area.

6.2 Suggestions for Future Research

There are many possible avenues of future research in automated RNN modeling. For instance, the RNN structure presented in this thesis utilizes only output feedback. It could be interesting to consider other RNN-type structures that contain more feedback pathways such as internal feedback in the hidden layer or even feedback in each neuron. Perhaps the additional feedback present in such RNN structures could lead to further model reduction and thereby a more compact model for a given training waveform set. However with more feedback, the training convergence may become problematic with

gradient-based methods based on the BPTT concept. Therefore an important related research topic is to investigate computationally efficient RNN training algorithms with superior convergence to gradient methods. A larger extension to this research is to consider a totally novel discrete-time dynamic NN model.

When AMG detects underlearning during RNN training, the solution is to increase the RNN by adding more hidden neurons or order. An optimal strategy to grow the structure when underlearning is detected is a useful direction to upgrade the automatic RNN modeling process. For certain training waveforms, adding more neurons (freedom) may have more benefit than increasing order (memory) and vice versa. Similarly, when AMG tries to reduce the structure after good learning, an efficient pruning algorithm should be developed to arrive at a compact model in a systematic manner. These research topics would help to speed up the automated development of TD models using the RNN.

Since the scope of this thesis is TD modeling, the stability of the RNN is an important criteria that should be enforced. Currently RNN stability can only be checked as a post-processing step after training. If the RNN is not stable, the structure has to be re-initialized and a new round of training must begin. This results in an increase in the RNN training period and further slows down the automated RNN modeling technique. Perhaps stable NN training adaptation laws can be developed so that the internal weights of the RNN are only changed in such a manner that the RNN remains globally stable during training. Such a research area would be very useful and have many applications in future EDA research.

Bibliography

- [1] Q. J. Zhang and K. C. Gupta, *Neural Networks for RF and Microwave Design*. Norwood, MA: Artech House, 2000.
- [2] Q. J. Zhang, K. C. Gupta, and V. K. Devabhaktuni, "Artificial neural networks for RF and microwave design—from theory to practice," *IEEE Trans. Microwave Theory & Tech.*, vol. 51, no. 4, pp. 1339-1350, April 2003.
- [3] J. E. Rayas-Sanchez, "EM-based optimization of microwave circuits using artificial neural networks: the state-of-the-art," *IEEE Trans. Microwave Theory & Tech.*, vol. 52, no. 1, pp. 420-435, January 2004.
- [4] A. H. Zaabab, Q. J. Zhang, and M. S. Nakhla, "A neural network modeling approach to circuit optimization and statistical design," *IEEE Trans. Microwave Theory & Tech.*, vol. 43, no. 6, pp. 1349-1358, June 1995.
- [5] H. Sharma and Q. J. Zhang, "Automated time domain modeling of linear and nonlinear microwave circuits using recurrent neural networks," *IEEE Trans. Microwave Theory & Tech.*, (to be submitted).
- [6] H. Sharma and Q. J. Zhang, "Transient electromagnetic modeling using recurrent neural networks," *2005 IEEE MTT-S Int. Microwave Symp. Dig.*, Long Beach, CA, June 2005.
- [7] J. W. Bandler, M. A. Ismail, J. E. Rayas-Sanchez, and Q. J. Zhang, "Neuromodeling of microwave circuits exploiting space-mapping technology,"

- IEEE Trans. Microwave Theory & Tech.*, vol. 47, no. 12, pp. 2417-2427, December 1999.
- [8] P. M. Watson and K. C. Gupta, "EM-ANN models for microstrip vias and interconnects in dataset circuits," *IEEE Trans. Microwave Theory & Tech.*, vol. 44, no. 12, pp. 2495-2503, December 1996.
- [9] X. Ding, V. K. Devabhaktuni, B. Chattaraj, M. C. E. Yagoub, M. Deo, Jianjun Xu, and Q. J. Zhang, "Neural-network approaches to electromagnetic-based modeling of passive components and their applications to high-frequency and high-speed nonlinear circuit optimization," *IEEE Trans. Microwave Theory & Tech.*, vol. 52, no. 1, pp. 436-449, January 2004.
- [10] P. M. Watson and K. C. Gupta, "Design and optimization of CPW circuits using EM-ANN models for CPW components," *IEEE Trans. Microwave Theory & Tech.*, vol. 45, no. 12, pp. 2515-2523, December 1997.
- [11] V. Rizzoli, A. Costanzo, D. Masotti, A. Lipparini, and F. Mastri, "Computer-Aided Optimization of Nonlinear Microwave Circuits With the Aid of Electromagnetic Simulation," *IEEE Trans. Microwave Theory & Tech.*, vol. 52, no. 1, pp. 362-377, January 2004.
- [12] A. Veluswami, M. S. Nakhla, and Q. J. Zhang, "The application of neural networks to EM-based simulation and optimization of interconnects in high-speed VLSI circuits," *IEEE Trans. Microwave Theory & Tech.*, vol. 45, no. 5, pp. 712-723, May 1997.

- [13] T. Horng, C. Wang, and N.G. Alexopoulos, "Microstrip circuit design using neural networks," *IEEE MTT-S Int. Microwave Symp. Dig.*, Atlanta, GA, 1993, pp. 413-416.
- [14] P. M. Watson, G. L. Creech, and K. C. Gupta, "Knowledge based EM-ANN models for the design of wide bandwidth CPW patch/slot antennas," *IEEE APS Int. Symp. Dig.*, Orlando, FL, July 1999, pp. 2588-2591.
- [15] C. Cho and K. C. Gupta, "EM-ANN modeling of overlapping open-ends in multiplayer microstrip lines for design of bandpass filters," *IEEE APS Int. Symp. Dig.*, Orlando, FL, July 1999, pp. 2592-2595.
- [16] A. H. Zaabab, Q. J. Zhang, and M. S. Nakhla, "Device and circuit-level modeling using neural networks with faster training based on network sparsity," *IEEE Trans. Microwave Theory & Tech.*, vol. 45, no. 10, pp. 1696-1704, October 1997.
- [17] F. Wang and Q. J. Zhang, "Knowledge-based neural models for microwave design," *IEEE Trans. Microwave Theory & Tech.*, vol. 45, no. 12, pp. 2333-2343, December 1997.
- [18] K. Shirakawa, M. Shimizu, N. Okubo, and Y. Daido, "A large-signal characterization of an HEMT using a multilayered network," *IEEE Trans. Microwave Theory & Tech.*, vol. 45, no. 9, pp. 1630-1633, September 1997.
- [19] K. Shirakawa, M. Shimizu, N. Okubo, and Y. Daido, "Structural determination of multilayered large-signal neural-network HEMT model," *IEEE Trans. Microwave Theory & Tech.*, vol. 46, no. 10, pp. 1367-1375, October 1998.

- [20] V.K. Devabhaktuni, C. Xi, and Q.J. Zhang, "A neural network approach to the modeling of heterojunction bipolar transistors from S-parameter data," *Proc. 28th European Microwave Conf.*, Amsterdam, Netherlands, Oct. 1998, pp. 306-311.
- [21] M. Vai and S. Prasad, "Qualitative modeling heterojunction bipolar transistors for optimization: A neural network approach," *Proc. IEEE/Cornell Conf. Adv. Concepts in High Speed Semiconductor Dev. and Circuits.*, 1993, pp. 219-227.
- [22] Y. Fang, M. C. E. Yagoub, F. Wang, and Q. J. Zhang, "A new macromodeling approach for nonlinear microwave circuits based on recurrent neural networks," *IEEE Trans. Microwave Theory & Tech.*, vol. 48, no. 12, pp. 2335-2344, December 2000.
- [23] Jianjun Xu, M. C. E. Yagoub, Runtao Ding, and Q. J. Zhang, "Neural based dynamic modeling of nonlinear microwave circuits," *IEEE Trans. Microwave Theory & Tech.*, vol. 50, no. 12, pp.2769-2780, December 2002.
- [24] V. K. Devabhaktuni, M. C. E. Yagoub, and Q. J. Zhang, "A robust algorithm for automatic development of neural-network models for microwave applications," *IEEE Trans. Microwave Theory & Tech.*, vol. 49, no. 12, pp.2282-2291, December 2001.
- [25] G. V. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Systems*, vol. 2, pp. 303-314, 1989.
- [26] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P. Glorennec, H. Hjalmarsson, and A. Juditsky, "Nonlinear black-box modeling in system

- identification: a unified overview,” *Automatica*, vol. 31, no. 12, pp. 1691-1724, December 1995.
- [27] J.Sjöberg, H.Hjalmarsson, and L.Ljung, *Neural networks in system identification*. Linköping, Sweden: Linköping Univ., Tech. Rep., 1993.
 - [28] D. M. M.-P. Schreurs, J. A. Jargon, K. A. Remley, D. C. DeGroot, and K. C. Gupta, “Artificial neural network model for HEMTs constructed from large-signal time-domain measurements,” *ARFTG Conference Digest*, Spring 2002, June 2002, pp. 31-36.
 - [29] Y. Pan, S. W. Sung, and J. H. Lee, “Nonlinear dynamic trend modeling using feedback neural networks and prediction error minimization,” *IFAC Symp. Proceedings (ADCHEM 2000)*, Pisa, Italy, June 2000, pp. 827-832.
 - [30] J. Choi, T. H. Yeap, and M. Bouchard, “Nonlinear state-space modeling using recurrent multiplayer perceptrons with unscented Kalman filter,” *Proc. of IEEE International Conf. on Systems, Man and Cybernetics (SMC 2004)*, vol. 4, pp. 3427-3432, The Hague, Netherlands, October 2004.
 - [31] L. Behera, S. Kumar, S. C. Das, “Identification of nonlinear dynamical systems using recurrent neural networks,” *TENCON 2003*, vol. 3, pp. 1120-1124, October 2003.
 - [32] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550-1560, October 1990.
 - [33] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157-166, March 1994.

- [34] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 279-297, March 1994.
- [35] N. E. Barabanov and D. V. Prokhorov, "Stability analysis of discrete-time recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 13, no. 2, pp. 292-303, March 2002.
- [36] M. H. Bakr, P. P. M. So, and W. J. R. Hoefer, "The Generation of Optimal Microwave Topologies Using Time-Domain Field Synthesis," *IEEE Trans. Microwave Theory & Tech.*, vol. 50, no. 11, pp. 2537-2544, November 2002.
- [37] F. T. Ulaby, *Fundamentals of Applied Electromagnetics*. Upper Saddle River, NJ: Prentice Hall, 2001.
- [38] S. Grivet-Talocia, "Package macromodeling via time-domain vector fitting," *IEEE Microwave and Wireless Components Letters*, vol. 13, no. 11, pp. 472-474, November 2003.
- [39] B. Gustavsen and A. Semlyen, "A robust approach for system identification in the frequency domain," *IEEE Trans. Power Delivery*, vol. 19, no. 3, pp. 1167-1173, July 2004.
- [40] Achar and M. S. Nakhla, "Simulation of high-speed interconnects," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 693-728, May 2001.
- [41] B. Gustavsen and A. Semlyen, "Rational approximation of frequency domain responses by vector fitting," *IEEE Trans. Power Delivery*, vol. 14, no. 3, pp. 1052-1061, July 1999.

- [42] V. Antonini, "SPICE equivalent circuits of frequency-domain responses," *IEEE Trans. Electromagnetic Compatibility*, vol. 45, no. 3, pp. 502-512, August 2003.
- [43] B. Gustavsen, "Computer code for rational approximation of frequency dependent admittance matrices," *IEEE Trans. Power Delivery*, vol. 17, no. 4, pp. 1093-1098, October 2002.
- [44] B. C. Kuo, *Automatic Control Systems*, 7th Edition, New York City, NY: John Wiley & Sons Inc., 1995.
- [45] W. J. R. Hoefer and P. P. M. So, *The MEFiSTo-2D Theory*, Victoria, BC, Canada: Faustus Scientific Corporation, 2001.
- [46] D. A. Frickey, "Conversions between S, Z, Y, h, ABCD, and T parameters which are valid for complex source and load impedances," *IEEE Trans. Microwave Theory & Tech.*, vol. 42, no. 2, pp. 205-211, February 1994.
- [47] B. Gustavsen and A. Semlyen, "Enforcing passivity for admittance matrices approximated by rational functions," *IEEE Trans. Power Systems*, vol. 16, no. 1, pp. 97-104, February 2001.
- [48] *MEFiSTo-3D Pro*, Version 4.0, Faustus Scientific Corp., Victoria, BC, 2005.
- [49] J. Wood and D. E. Root, *Fundamentals of Nonlinear Behavioral Modeling for RF and Microwave Design*. Norwood, MA: Artech House, 2005.
- [50] D. Wisell, "A baseband time domain measurement system for dynamic characterization of power amplifiers with high dynamic range over large bandwidths," presented at IMTC 2003, Vail, CO, May 2003.

- [51] T. Liu, S. Boumaiza, and F. M. Ghannouchi, "Dynamic behavioral modeling of 3G power amplifiers using real-valued time-delay neural networks," *IEEE Trans. Microwave Theory & Tech.*, vol. 52, no. 3, pp. 1025-1033, March 2004.
- [52] M. Isaksson, D. Wisell, and D. Rönnow, "Nonlinear behavioral modeling of power amplifiers using radial-basis function neural networks," *2005 IEEE MTT-S Int. Microwave Symp. Dig.*, Long Beach, CA, June 2005.
- [53] *Agilent-ADS*, Version 2003a, Agilent Technologies, Santa Rosa, CA, 2003.